COLLEGE CODE: 9233

COLLEGE NAME: GOVERNMENT COLLEGE OF ENGINEERING BODINAYAKUNUR

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

STUDENT_ID: 8AE480233C809F1A2D54F980ABF4887C

REGISTER NO: 923323104305

DATE:24.10.2025

COMPLETED THE PROJECT NAMED AS PHASE - 5

TECHNOLOGY PROJECT NAME: REAL TIME CHAT BOT

SUBMITTED BY

NAME: THIRUPATHI P

MOBILENO: 9345110398

Final Demo Walkthrough:

A "Real-Time Chatbot Final Demo Walkthrough" project title refers to a presentation or demonstration showcasing the completed development of a real-time chatbot system. It typically includes demonstrating the chatbot's ability to handle live interactions, respond instantly to user queries, integrate with existing systems, and provide real-time insights or reporting.

Key elements to cover in such a demo walkthrough include:

- Introduction to the chatbot's purpose and key features.
- Live demonstration of the chatbot responding to real-time user inputs.
- Explanation of the underlying technology such as speech-to-text transcription if applicable, or AI models driving the chatbot.
- Integration with messaging channels or web platforms.
- Testing and iteration phases highlighting how the bot was refined.
- Deployment options and how users can access the chatbot (e.g., via URL, widget, or messaging apps).
- Insights on scalability and adaptability to different business or conversational needs.

For example, a walkthrough could start with showing how the chatbot handles complex questions seamlessly, demonstrating real-time conversation streaming, and concluding with deployment details and potential use cases for business automation or customer service enhancement.

This approach ensures the audience understands both the technical and practical aspects of the chatbot, reinforcing its value and readiness for production us.

Project Report:

A real-time chatbot project report typically involves designing and implementing a conversational AI system that interacts with users in real time. Such chatbots use natural language processing (NLP) and machine learning techniques to understand and respond to user queries instantly, simulating human-like conversation through text or voice interfaces.

Architecture Overview:

1. Frontend: React + TallwindCSS
2. Backend: Node.js + Express
3. Database: Xampp

Development Phases

Phase 1: Problem Understanding & Requirements

- Analyze the current situation and gather data about the business context, existing processes, and customer pain points.
- Define the chatbot’s main objective as a SMART goal (Specific, Measurable, Achievable, Relevant, Time-bound).

Phase 2: Solution Design & Architecture

- Design conversation flows, user interactions, and overall chatbot architecture ensuring empathy and addressing fallback situations.
- Define how the chatbot handles user inputs, maintains context, and connects to knowledge bases.

Phase 3: MVP Implementation

- Develop a Minimum Viable Product (MVP) that covers core use cases such as FAQs, simple guidance, or handover to human agents.
- Build the backend architecture and integrate the necessary APIs and NLP/AI models.

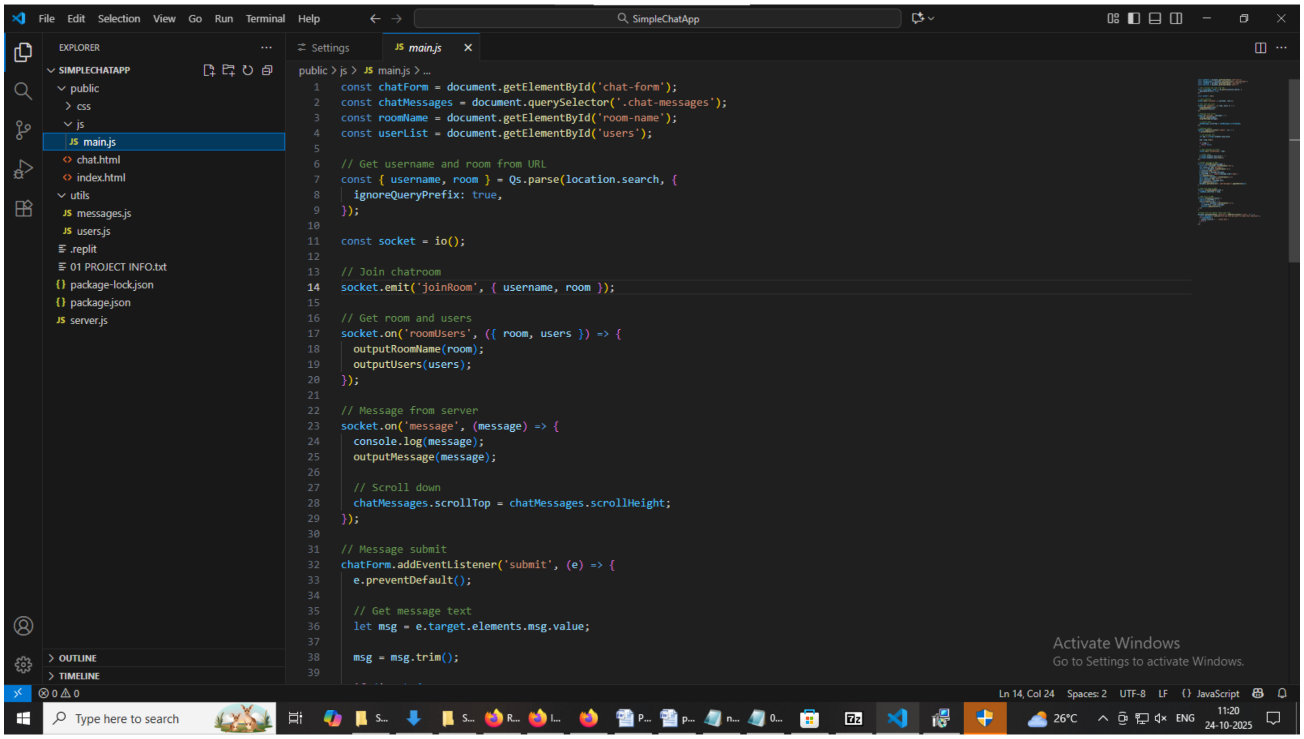
Phase 4: Enhancements & Deployment

- Incorporate AI enhancements including advanced NLP, machine learning for continuous improvement, and sentiment analysis.
- Expand functionality by adding more conversation paths and integrations.

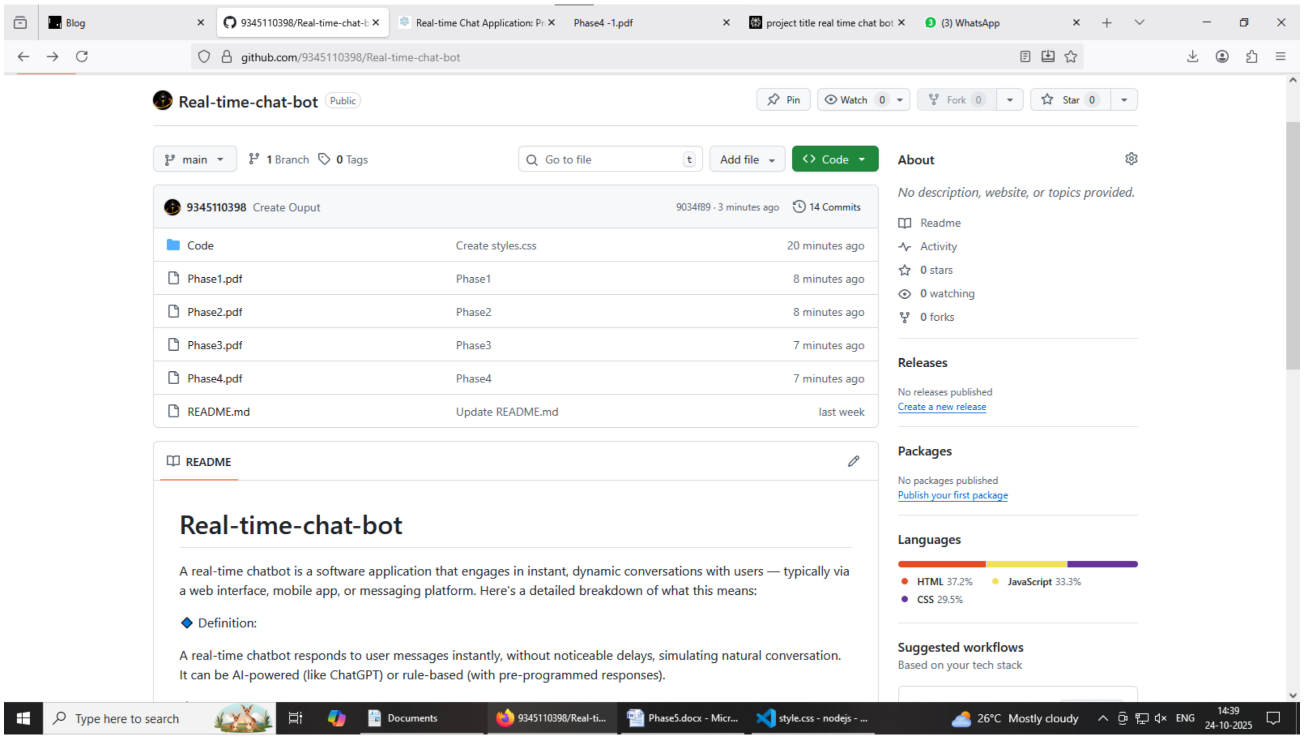
Phase 5: Project Demonstration & Documentation

- Present the fully functional chatbot solution to stakeholders highlighting achievements against initial KPIs and goals.
- Provide detailed documentation including system architecture, design decisions, API integrations, user guides, and maintenance plans.
- Include lessons learned and recommendations for future improvements or scaling [general best practices].

Screenshots and API Documentation:



```javascript
const chatForm = document.getElementById('chat-form');
const chatMessages = document.querySelector('.chat-messages');
const roomName = document.getElementById('room-name');
const userList = document.getElementById('users');

// Get username and room from URL
const { username, room } = Qs.parse(location.search, {
  ignoreQueryPrefix: true,
});

const socket = io();

// Join chatroom
socket.emit('joinRoom', { username, room });

// Get room and users
socket.on('roomUsers', ({ room, users }) => {
  outputRoomName(room);
  outputUsers(users);
});

// Message from server
socket.on('message', (message) => {
  console.log(message);
  outputMessage(message);

  // Scroll down
  chatMessages.scrollTop = chatMessages.scrollHeight;
});

// Message submit
chatForm.addEventListener('submit', (e) => {
  e.preventDefault();

  // Get message text
  let msg = e.target.elements.msg.value;

  msg = msg.trim();

```



9345110398 Create Ouput — 9034f89 · 3 minutes ago — 14 Commits

Code	Create styles.css	20 minutes ago
Phase1.pdf	Phase1	8 minutes ago
Phase2.pdf	Phase2	8 minutes ago
Phase3.pdf	Phase3	7 minutes ago
Phase4.pdf	Phase4	7 minutes ago
README.md	Update README.md	last week

README

Real-time-chat-bot

A real-time chatbot is a software application that engages in instant, dynamic conversations with users — typically via a web interface, mobile app, or messaging platform. Here's a detailed breakdown of what this means:

🔷 Definition:

A real-time chatbot responds to user messages instantly, without noticeable delays, simulating natural conversation. It can be AI-powered (like ChatGPT) or rule-based (with pre-programmed responses).

About

No description, website, or topics provided.

Readme
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

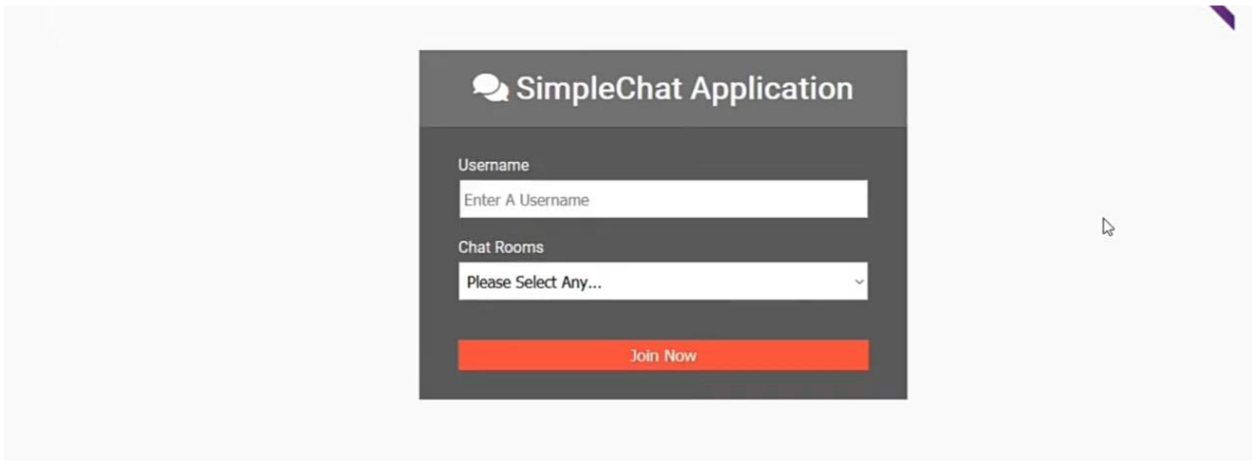
No packages published
Publish your first package

Languages

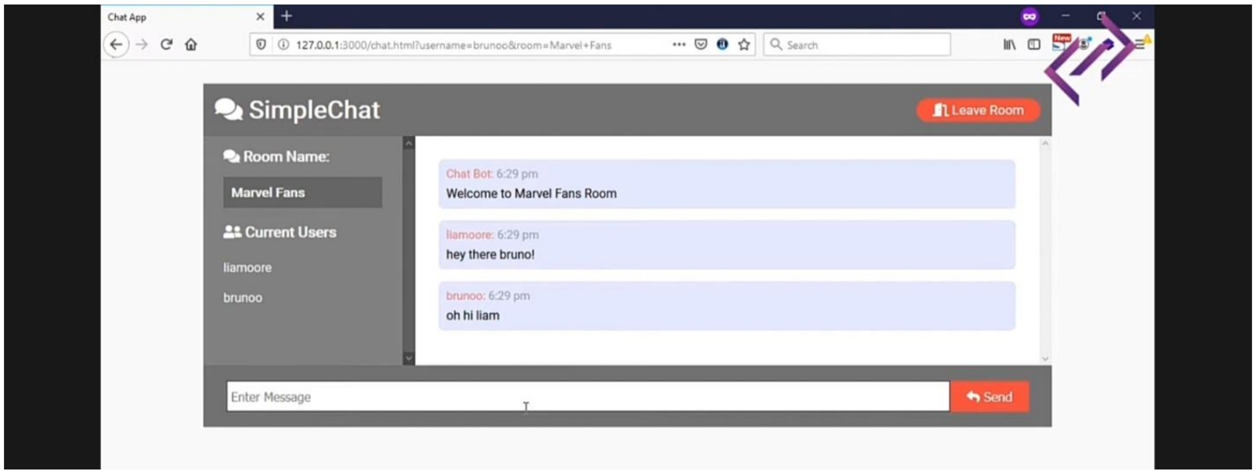
● HTML 37.2% ● JavaScript 33.3% ● CSS 29.5%

Suggested workflows
Based on your tech stack

Login page:



Chat main menu:

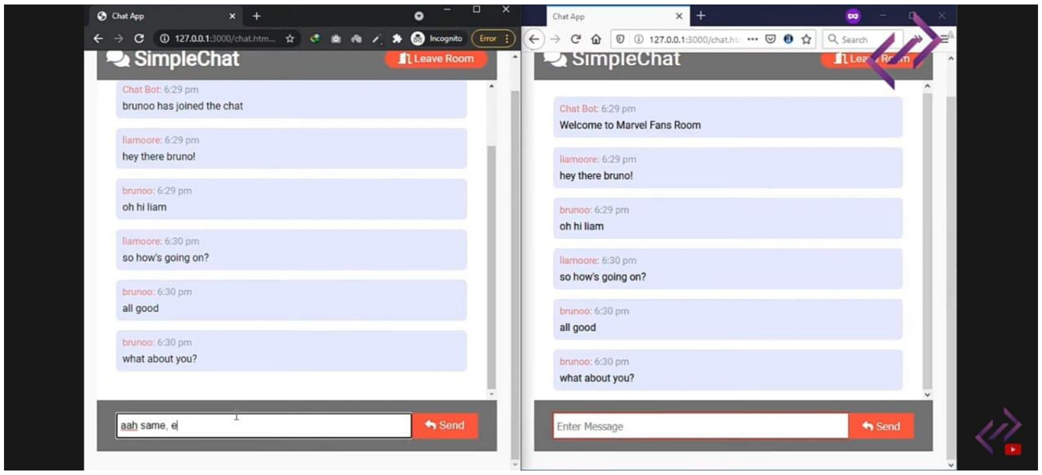


Chat Bot: 6:29 pm
Welcome to Marvel Fans Room

liamoore: 6:29 pm
hey there bruno!

brunoo: 6:29 pm
oh hi liam

Chat rooms:



Chat Bot: 6:29 pm
brunoo has joined the chat

liamoore: 6:29 pm
hey there bruno!

brunoo: 6:29 pm
oh hi liam

liamoore: 6:30 pm
so how's going on?

brunoo: 6:30 pm
all good

brunoo: 6:30 pm
what about you?

aah same, e

Chat Bot: 6:29 pm
Welcome to Marvel Fans Room

liamoore: 6:29 pm
hey there bruno!

brunoo: 6:29 pm
oh hi liam

liamoore: 6:30 pm
so how's going on?

brunoo: 6:30 pm
all good

brunoo: 6:30 pm
what about you?

1. **Chat Interface Screen**
    a. Shows a live chat window where users type messages and receive immediate responses from the bot.Displays conversation flow with timestamps and message roles (User/AI).
2. **Message Streaming Status Screen**
    a. Shows the chatbot processing state such as "typing", "loading", or "error".Includes buttons to stop ongoing message generation and input fields disabled/enabled based on status.
3. **Chat History and Metadata Screen**
    a. Displays past chat messages with user and AI roles.Includes conversation metadata like token usage or response time shown underneath messages.

API Documentation:

Endpoint	HTTP Method	Description	Request Body	Response
`/api/chat`	POST	Send user message and receive AI response	`{ "messages": [{ "role": "user", "text": "Hello" }] }`	Returns AI-generated message stream (JSON)
`/api/chat/status`	GET	Get current chat session status	None	`{ "status": "ready" / "streaming" / "error" }`
`/api/chat/stop`	POST	Stop current message generation	None	`{ "success": true, "message": "Generation stopped" }`
`/api/chat/history`	GET	Retrieve past chat messages	None	`{ "messages": [{ "role": "user", "text": "...", "timestamp": "..." }, ...] }`

Challenges And Solutions:

Challenge	Description	Solution	Typical HTTP Methods
Integration with Existing Systems	Difficulty connecting chatbot with backend CRM, ERP, or legacy APIs, causing slow or failed data flow	Use robust API-driven integration and middleware to bridge legacy systems; involve IT early	GET (fetch data), POST (submit user data), PUT/PATCH (update data), DELETE (remove data)
Maintaining Real-time Performance	Handling multiple simultaneous user requests without lag or downtime	Use scalable cloud infrastructure, load balancing, and continuous performance monitoring	GET (for fetching responses), POST (for sending user input)
Ensuring Data Security and Privacy	Handling sensitive customer data risks data breaches and compliance violations	Implement encryption, access controls, multi-factor authentication, and comply with legal standards	POST (securely transmit data), HTTPS to secure all methods

Git Hub Respository:

Link : https://github.com/9345110398/Real-time-chat-bot.git